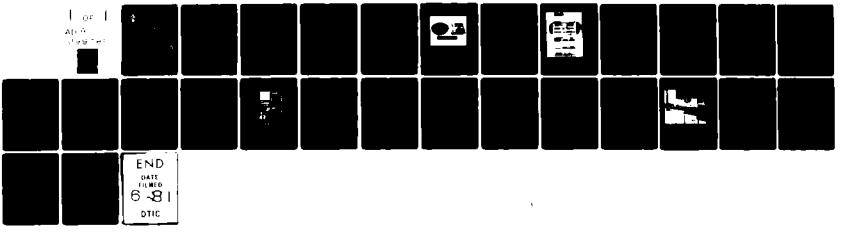
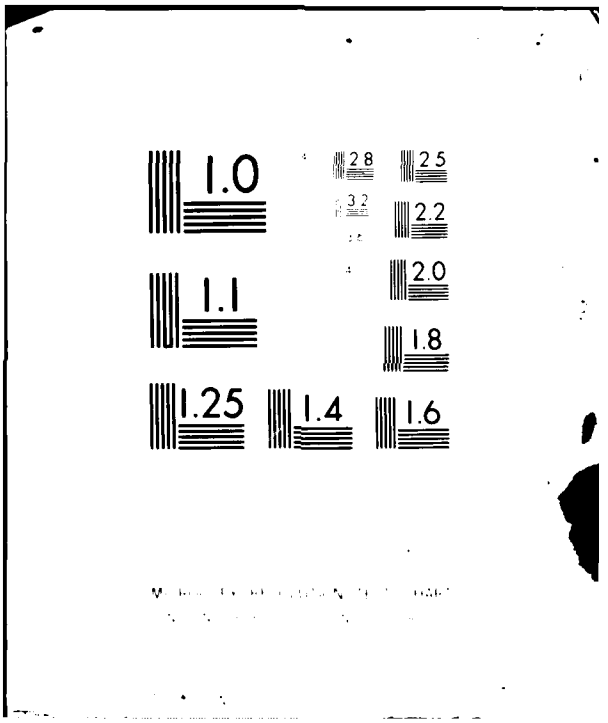
AD-A099 785 ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) F/G 9/2
LECTURE NOTES ON A MACHINE INDEPENDENT COMPILER FOR ATLAS (MICA--ETC(U)
OCT 80 D P TAYLOR
UNCLASSIFIED RSRE-MEMO-3310 DRIC-BR-77228 NL



END
DATE FILMED
6-81
DTIC

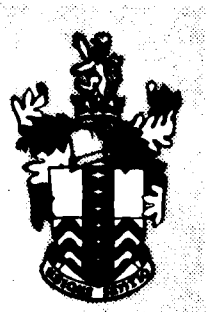

1.0
2.8
2.5
3.2
2.2
1.1
2.0
1.8
1.25
1.4
1.6

BR 77228

UNLIMITED

LEVEL II

(1)



AD A099785

RSRE
MEMORANDUM No. 3310

ROYAL SIGNALS & RADAR ESTABLISHMENT

LECTURE NOTES ON A MACHINE INDEPENDENT COMPILER FOR ATLAS (MICA)

Author: D P Taylor

DTIC ELECTE JUN 0 5 1981 S D E

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

This document is the property of Procurement Executive, Ministry of Defence. Its contents should not be made public either directly or indirectly without approval from HBM Secretary of State for Defence (Director RSRE) .

RSRE MEMORANDUM No. 3310

DTIC FILE COPY

81 6 03 077

(18) DRIC / (19) BR-77228

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3310

Title: (6) LECTURE NOTES ON A MACHINE INDEPENDENT COMPILER FOR ATLAS (MICA)

Author: (10) D. P. Taylor

Date: (11) Oct [redacted]80 (12) 28 (14) RSRE-MEMO-3310

SUMMARY

(Memorandum repl.)

The memorandum is in essence the text of a lecture given to the Industrial Liaison Group of the Defence Automatic Test Equipment Steering Committee, on 10 September 1980 at RSRE Malvern. The purpose of the lecture was to convey to industry the results of research work performed by RSRE into the derivation of a Machine Independent Compiler for ATLAS (MICA).

Accession For
NTIS GRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

This memorandum is for advance information. It is not necessarily to be regarded as a final or official statement by Procurement Executive, Ministry of Defence

Copyright
C
Controller HMSO London
1980

409929 slk

RSRE MEMORANDUM NO 3310

LECTURE NOTES ON A MACHINE INDEPENDENT COMPILER FOR ATLAS(MICA)

D P Taylor

CONTENTS

1 INTRODUCTION

I have partitioned my lecture on aspects of MICA which are of particular relevance to industry into four brief talks. The first talk called "What is MICA and what can be made available to industry", will initially introduce the ATLAS language thus setting MICA into context. I will then describe, in sketch form, what MICA is. Following on from this I will describe what can be made available to industry and finally what can not be made available to industry. The second talk concerns itself with an outline of the MICA pre-compiler and the third talk covers the post-compiler aspects of MICA. The final talk called "MICA the final report" is an overview of the acceptance work which we at RSRE have performed on MICA. I have included this aspect of the work because it will highlight the magnitude of MICA and will also demonstrate that MICA is not a proven product which can be used with total confidence.

2 TALK 1. WHAT IS MICA AND WHAT CAN BE MADE AVAILABLE TO INDUSTRY

What is MICA and what can be made available to industry? Before delving into the subject matter I believe that it is appropriate at this stage to give a very brief overview of the Abbreviated Test Language for all Systems (ATLAS).

2.1 Overview of ATLAS

ATLAS was originally developed for avionics applications under the auspices of Aeronautical Radio Inc. (ARINC) and the first version was approved in 1968. The purpose of ATLAS is to provide a standardised test language for expressing test specifications and procedures which are independent of the target test equipment. In 1976, when ATLAS was approved as an IEEE standard, thirteen language revisions had occurred and the reprint of the ARINC revision, known as 13A was given the IEEE standard 416-1976. The MICA work described today is based on a subset of IEEE standard 416-1976.

Unfortunately, ATLAS has to date had one striking aspect and that is its distinct lack of popularity, although most bodies involved with testing and ATE all agree that considerable benefits to manufacturers and users alike can accrue from the use of a standard high-level test orientated language. To gain an appreciation of the reasons for ATLAS's lack of popularity consider the following. The ATLAS language controlling body consists of voluntary experts and there has throughout the history of the language been a remarkable lack of main purpose, in particular this being whether ATLAS is an ATE programming language or simply a test specification language. Because to make such a decision by voluntary committee is difficult and because few computer language/compiler experts have had any serious involvement in the early language definition stage, a vague compromise position has been reached. In other words ATLAS has not been devised to simplify the problems of converting (compiling) ATLAS programs into ATE machine code. Put another way the job of deriving ATLAS compilers has, to date, proved to be too complex and expensive to encourage ATE manufacturers to supply an ATLAS compiler as part of their product range. Therefore it is not surprising that if ATLAS programs can not be compiled the inducement to write in ATLAS at all is not commercially attractive. A further glaring problem with ATLAS is its deficiency as a digital test language, although considerable efforts are being made at the present time to improve the situation, we are still some way off an agreed and implemented solution.

Having recognised and highlighted the ATLAS problem areas we in MOD are still convinced of the value of using an unambiguous high-level test specification language and MOD's aspirations are embodied in the document "A Guide For The Defence Industry In The Use Of ATLAS" Defence Standard 00-14. Furthermore we at RSRE hope that the results of our MICA research work will encourage industry to look perhaps a little more carefully at the use of ATLAS in MOD contracts.

2.2 What is MICA

What then is MICA? MICA can be considered as a base line ATLAS compiler which can potentially be modified in a variety of ways to suit particular applications. MICA in its base line form will accept as input an ATLAS program which conforms to a 60% subset of ATLAS 416-1976. It will grammatically analyse the ATLAS program and if correct will output a coded representation of ATLAS. Following on from this further analysis takes place which relates the physical measurement and stimuli resources available on an ATE selected for the research work, to the coded ATLAS program and finally machine code for the ATE is produced.

(An example of a modification to the MICA base line is one that we have devised at RSRE which produces code for an experimental IEEE-488 bus ATE.)

Throughout the MICA research work considerable emphasis was placed on the requirement for easing the problems of compiler reconfiguration. This emphasis has been reflected into the philosophy and design of MICA and some of the primary features which should permit reconfiguration are:

i To ease the problems associated with the requirement that MICA should be able to run on a variety of computers MICA was written in MOD's general purpose, problem orientated language, CORAL 66.

As CORAL 66 compilers exist for the majority of popular computers used in this country the problems of transporting MICA, at least in theory, are considerable reduced.

ii A distinct separation in design and implementation has been made between the ATLAS grammatical analysis phase, known as the pre-compiler, and the ATE resource allocation and code generation phase which is known as the post-compiler.

iii To cater for the implementation of different subsets of ATLAS, to meet specific testing problems, the pre-compiler is driven from from a data base which is produced by semi-automatic methods.

iv To ensure that the detailed information concerning MICA was not buried in a highly complex mass of program listings a rigorous documentation standard was employed.

To summarise, MICA is a computer program written in CORAL 66 which will accept an ATLAS program as data and will produce machine code for the ATE selected for the research work.

2.3 What Can Be Made Available to Industry

I will now turn to the important area of what aspects of MICA we at RSRE can make available to industry.

MICA is the result of extra-mural research work performed for RSRE by Warren Point Ltd, under two separate, but technically linked contracts. Under the conditions of these contracts Warren Point Ltd have the Industrial Property Rights in all work performed under these contracts. However MOD has the right to a free licence to copy and use for United Kingdom Government purposes, all work, the Industrial Property Rights of which rest with Warren Point, as a result of the contracts. In other words industry can only make free use of MICA for United Kingdom Government purposes. So in the context of the use of MICA for United Kingdom Government purposes we at RSRE can provide a variety of software and documentation. I must however emphasise that all software and documentation is to be treated as Commercial-in-Confidence.

What we have done is to separate the MICA software and documentation into two separate levels. The first level we have termed "MICA Evaluation Documentation and Software" and the second level as a "Complete Set of MICA Documentation and Software". Physically level one is shown in Figure 1.

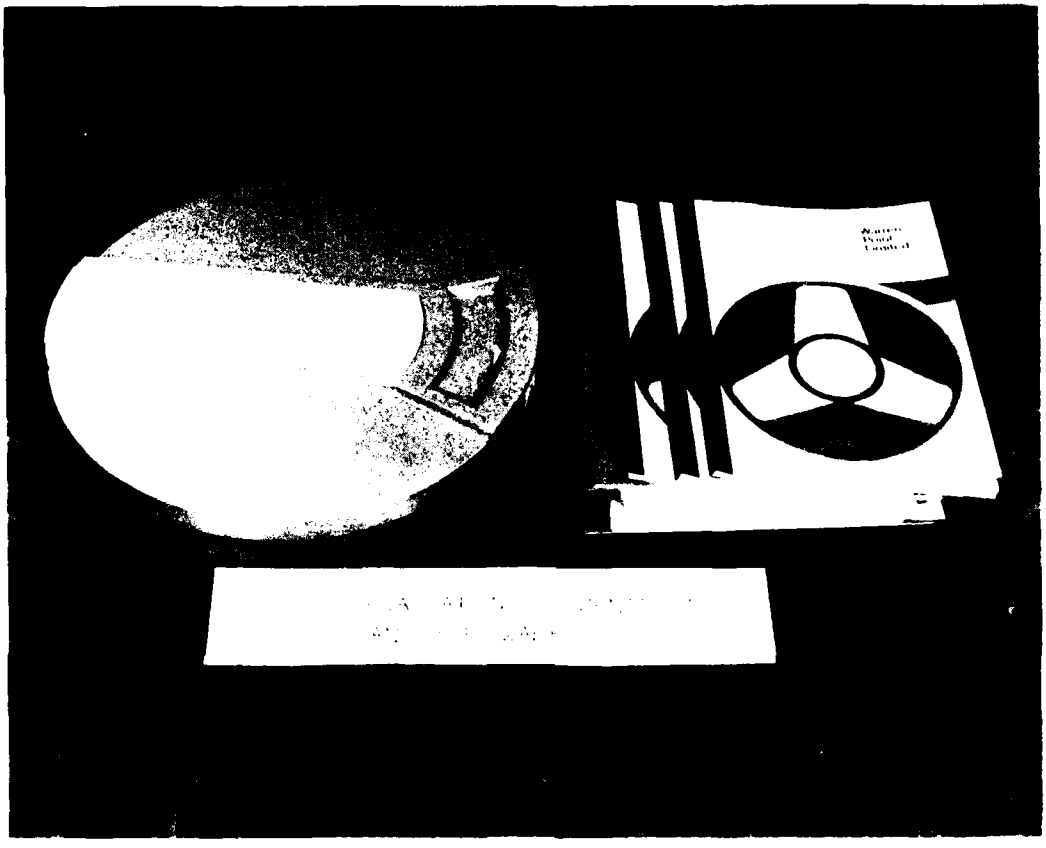


FIGURE 1

Level 1 comprises:

i A PDP 11 RKØ5 disc

and ii Four documents

The disc contains:

i MICA pre-compiler in a PDP 11 executable task form

ii MICA post-compiler in a PDP 11 executable task form

iii A definition of the intermediate code produced by the pre-compiler

and iv Various sample ATLAS programs.

The documents are:

i A definition of the RSRE subset implemented by the pre-compiler

ii A list of pre-compiler error report numbers

iii A MICA user's guide

and iv A code of practice for writing ATLAS for the post-compiler.

Thus level one is a complete package which will permit MICA to be run and evaluated on potentially any of the PDP 11 range computers which have a multi-user RSX 11M operating system.

Moving on to the level two documentation which is physically shown in Figure 2.

Level two is a complete set of MICA documentation and software and consists of full design specifications and software documentation for:

i The MICA pre-compiler

and ii The MICA post-compiler

The documentation is in accordance with the MOD documentation standard AVP 70 Spec 4 althouth there are certain deviations from the standard which permitted Warren Point Ltd to make use of a software based documentation method known as Program Design Language (PDL). (A list of documents at levels one and two is given in Appendix A).

2.4 What Can Not Be Made Available to Industry

The final aspect of this first talk is what aspects of MICA can not be made available to industry by RSRE. Figures 1 and 2 give an indication of the physical volume of MICA related information that is available. Not only is the information large in volume but it describes a highly complex piece of software and therefore would require a considerable effort, even with the documentation standards employed, to gain sufficient understanding to enable software changes to be made to MICA. Bearing this point in mind we at RSRE are not in a position to provide any further assistance with

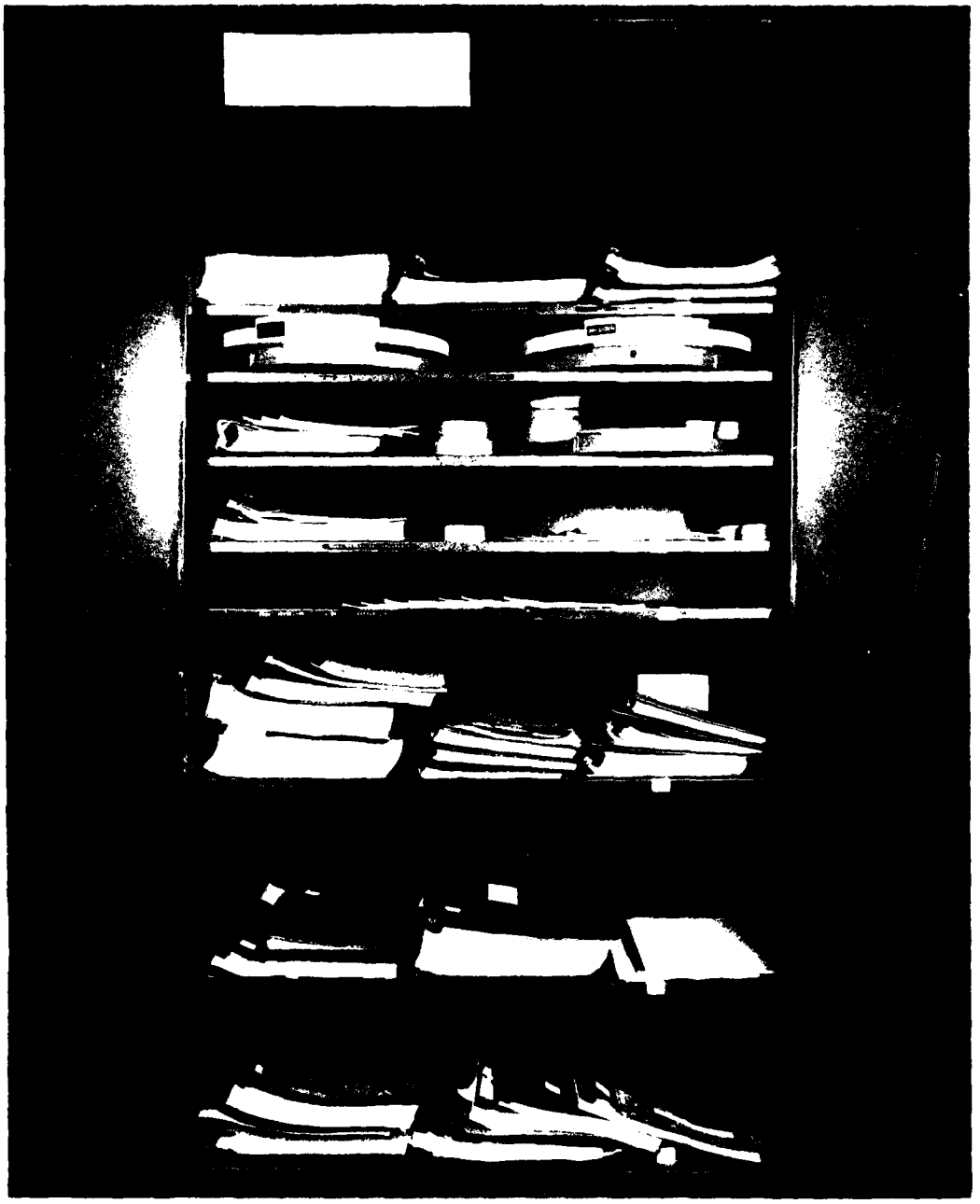


FIGURE 2

MICA to industry other than to issue the documentation and software described. It is not because we do not want to assist but rather that we simply do not have the available effort.

3 TALK 2. BRIEF DESCRIPTION OF THE MICA PRE-COMPILER

Moving on now to the second talk in which I will give an outline description of the MICA pre-compiler.

I have already indicated that one of the MICA research aims was to assist ATLAS compiler reconfigurability by splitting MICA into two very distinct parts, a pre-compiler for performing grammatical analysis and a post compiler for performing all the processing specific to a target ATE, by which I mean the allocation of ATE resources to ATLAS statements and ATE code generation. This two part approach is shown in Figure 3.

MACHINE INDEPENDENT COMPILER FOR ATLAS (MICA)

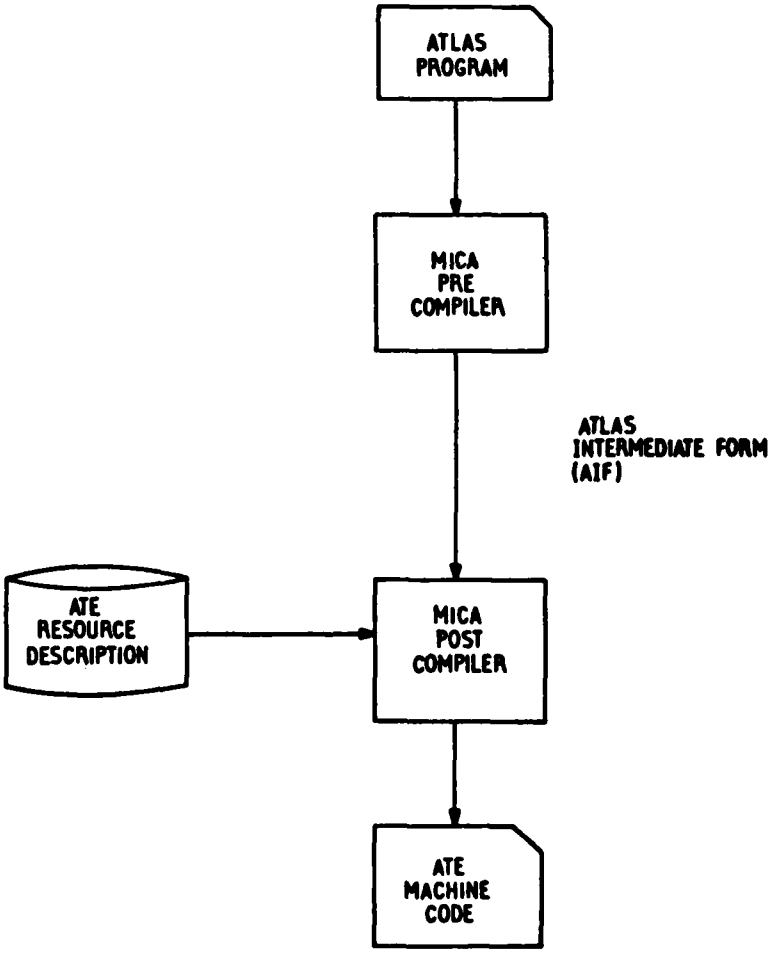


FIG.3

At the top of Figure 3 we have the input to MICA, an ATLAS program. The program is grammatically analysed by the pre-compiler and any program errors are reported to the operator. If the pre-compilation is successful then the output data from the pre-compiler is input to the post-compiler. The post compiler analyses this data and if resource errors exist then an error report is provided to the operator. Finally an ATE object program, equivalent to the ATLAS program is produced, in other words the ATE machine code.

In order to gain an appreciation of the functions performed by the pre-compiler it would be useful to look first at the form of a hypothetical ATLAS input statement shown in Figure 4 and then a sample ATLAS program

HYPOTHETICAL ATLAS STATEMENT

```
MEASURE, (FREQ), AC SIGNAL, FREQ MAX 32 MHz, VOLTAGE MAX 1 MV,
                 CNX HI A1 LO A2$
```

Figure 4

In Figure 4 we have the form of a hypothetical signal procedural statement. The first statement characteristic shown here is the action part, the verb which in this case is MEASURE. Following on from the verb the signal characteristic to be measured is enclosed in brackets. In this case the characteristic to be measured is frequency. Following this is the statement noun which is AC SIGNAL and this is followed by the statement characteristics associated with the noun. In this case two of many possible statement characteristics have been used. The final part of the statement is the connection field and indicates across which unit under test pins the measurement is to be taken.

SAMPLE ATLAS PROGRAM

```
E000000   BEGIN, ATLAS PROGRAM 'TEST' $
C     $
     05   DECLARE, DECIMAL, 'DUMMY' $
C         $
     10   DISPLAY, MESSAGE, CONNECT DVM TO BUS, CONNECT RES TO BE MEASURED
          PRESS GO WHEN READY $
C         $
     15   WAIT FOR, MANUAL INTERVENTION $
C         $
     20   MEASURE, (RES), IMPEDANCE, RES MAX 1ØKOHM,
          CNX H1 J1-1 LO J1-2 $
```

```
    24   DISPLAY, MESSAGE, MEASURED RESISTANCE = $
    26   DISPLAY, RESULT, 'MEASUREMENT' $
    30   DISPLAY, MESSAGE, KILOHMS $
000255   REMOVE, ALL $
000265   FINISH $
000270   TERMINATE, ATLAS PROGRAM 'TEST' $
```

Figure 5

The above ATLAS program, Figure 5, simply:

i Outputs a message to the ATE operator. Program step number 10.

ii Waits for a manual intervention when the ATE has been set up. Program step number 15

iii Performs a resistance measurement. Program step number 20.

iv Displays the result to the operator. Program step number 26, and finally

v Removes all ATE to UUT connections. Program step number 255.

You will notice that the program statements are numbered in ascending order and preceding the statement number is an optional flag field. Two flags used in this program are the "C" flag for commentary and "E" flag for entry point.

ATLAS is in fact an exceptionally wordy language. For example it has in the order of thirty-nine different nouns with one hundred and forty seven noun modifier variations. Therefore, you will appreciate that the example program shown is very limited. The sheer size of ATLAS in terms of number of reserved words and grammar makes the task of grammatical analysis a far from trivial matter.

Let us now consider what it is that the pre-compiler must do in order to grammatically analyse an ATLAS program.

The pre-compiler performs four major tasks when processing an ATLAS program, macro expansion, lexical analysis, syntax analysis and semantic analysis.

i The first task is the macro expansion of any macros involved in the program and production of a text expanded program. (ATLAS macros are of the DEFINED FUNCTION or DEFINED MESSAGE form).

ii The second task is a lexical analysis of each character of individual lines of the text expanded program. The lexical analyser removes redundancies, such as comments, and verifies that each language word used in a statement is permitted. It does this by cross reference to a table which contains all permitted language words.

iii The third task is a syntactic analysis of each statement. The syntax analyser makes use of a data base which is derived from the definition of ATLAS. Effectively the syntax analyser determines whether or not the construction of statement conforms to the rules of ATLAS.

iv The fourth task is the semantic analysis of the complete ATLAS program. The semantic analyser verifies that the meaning of each statement, with relation to the other program statements, make sense. For example if a label has been used in the program, has it been defined? When the semantic analysis is complete then a coded representation, known as ATLAS Intermediate Form (AIF), of the ATLAS program is produced.

Each of these tasks, except macro expansion, has access to a data base which, because of the design of MICA, can potentially be modified to suit particular ATLAS subsets. The actual data bases as used by MICA together with the tasks described form, what I have called, the pre-compiler MICA body. Whereas the source form of the tables and the way in which they are manipulated to form the data bases are external to the MICA body but are still constituent parts of the MICA philosophy.

So let us look in a little more detail at the functions of the MICA pre-compiler which is capable of grammatically analysing a 60% subset of ATLAS 416-1976. In Figure 6 we see the ATLAS program as input and the coded representation of that program output in ATLAS intermediate Form.

MICA CONSTITUENT PARTS

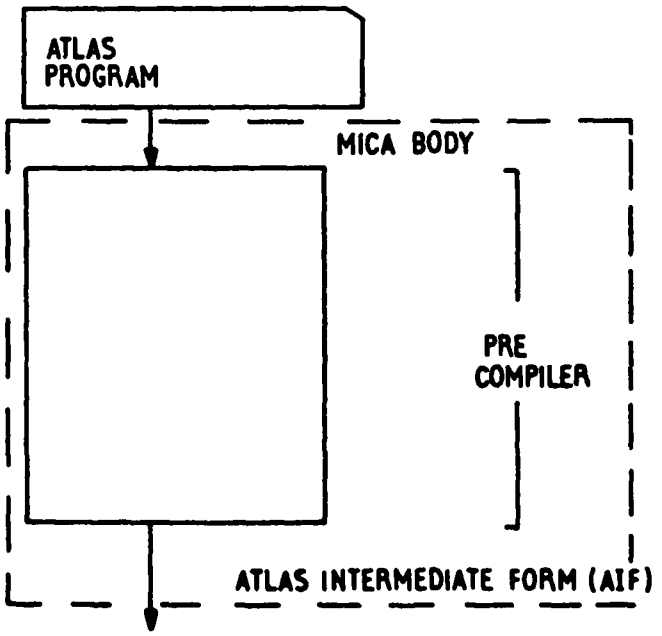


Figure 6

The first task that the pre-compiler performs is, as Figure 7 shows, macro expansion. It can be regarded as a pre-pass process which, as I have described, simply produces a text expanded ATLAS program.

MICA CONSTITUENT PARTS

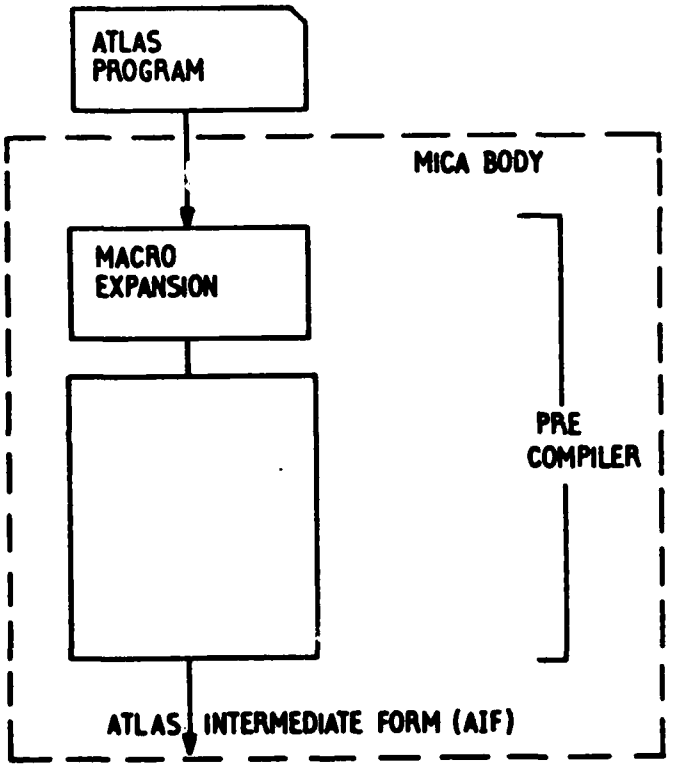


Figure 7

The ATLAS program is now processed by the lexical analyser on a character by character basis, as shown in Figure 8. The lexical analyser's data base, shown external to the MICA body, is derived from a table of ATLAS symbols used.

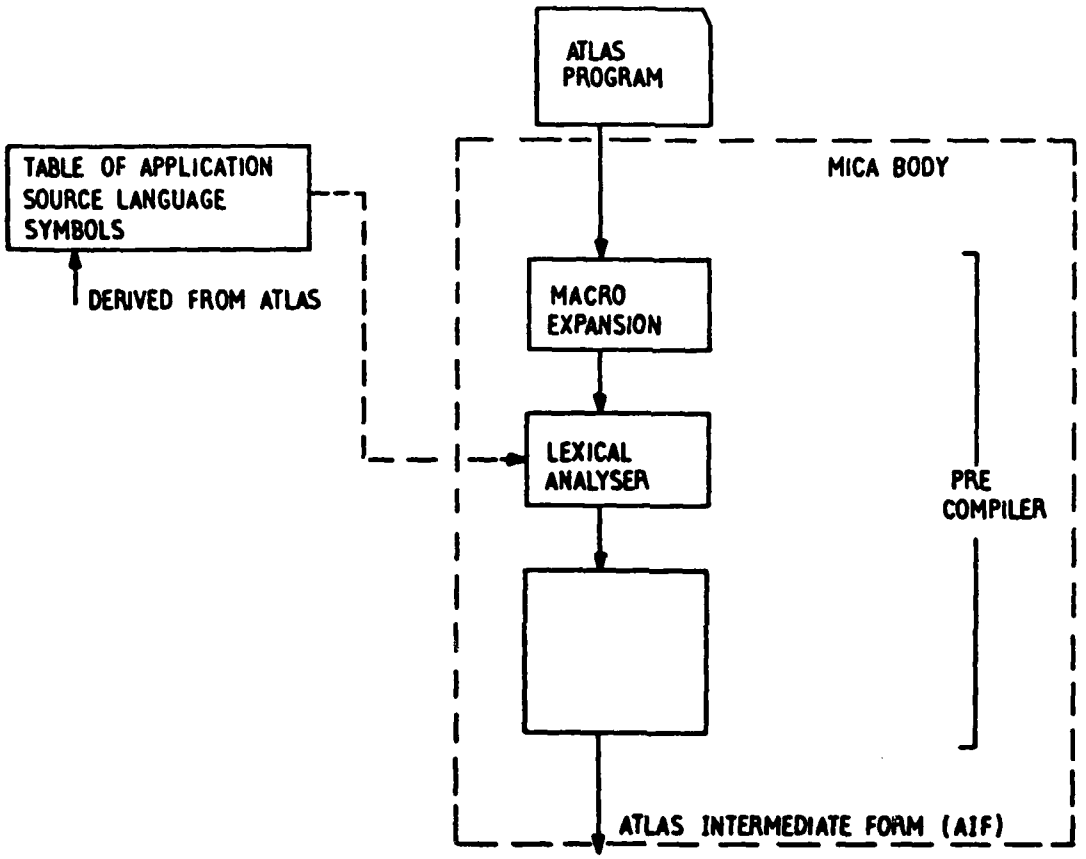


Figure 8

The major actions of the lexical analyser are:

i Symbol processing in which a symbol is built from the ASCII characters of the ATLAS program and if alphanumeric, is checked against the lexical analyser table for a match. If there is no match, or if the symbol is not alphanumeric, or the first character is not alphabetic, then checks are done to see if the symbol is a separator, statement delimiter, number, or a punctuation character.

ii Label processing which the current symbol is checked to be of the correct format for a label.

iii Comment processing in which the text up to the next prime is taken as comment and is discarded and finally

iv A two character token is generated to represent the lexical element found.

If a symbol is found which is not legal ATLAS then the lexical analyser returns an indication of this to an error report file.

As I have described the syntax analyser, shown in Figure 9, effectively determines whether or not the construction of each ATLAS statement conforms to the structural rules. The major part of syntax analysis within MICA is performed using a standard interpreter, with associated data base generated automatically from a modified BNF definition of ATLAS (a BNF definition of ATLAS is given in Volume 1 of the IEEE specification). This automatic compiler building tool is known as SEMSID[1] and all the SEMSID/MICA work was done using a GEC 4080 computer at RSRE Malvern. Much of the processing associated with the detection and reporting of syntactic program errors is also performed by SEMSID and to achieve this action calls are inserted at various key positions in the data base.

MICA CONSTITUENT PARTS

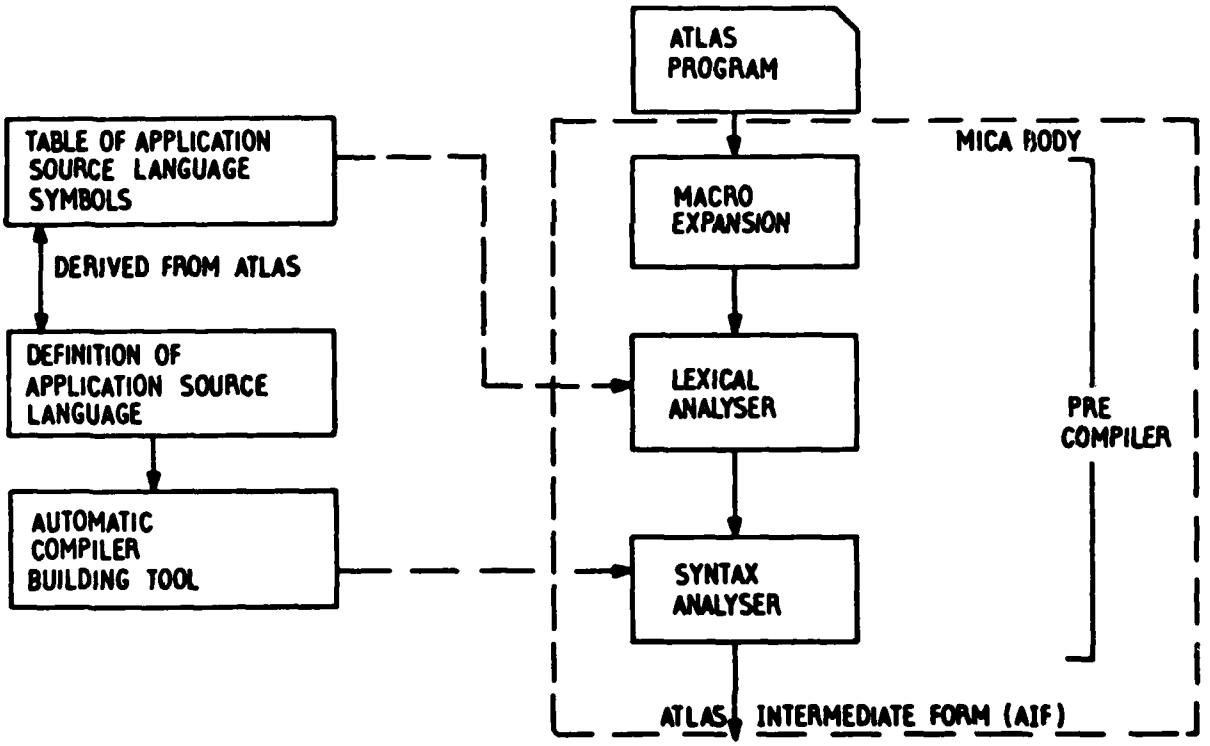


Figure 9

The final aspect of pre-compilation is semantic analysis and the generation of a coded form of the ATLAS program in AIF. This is shown in Figure 10. The semantic analysis is in part performed using the SEMSID data base and in part by other methods. When the semantic analyser has determined that the meaning of the ATLAS program is acceptable it then generates AIF.

MICA CONSTITUENT PARTS

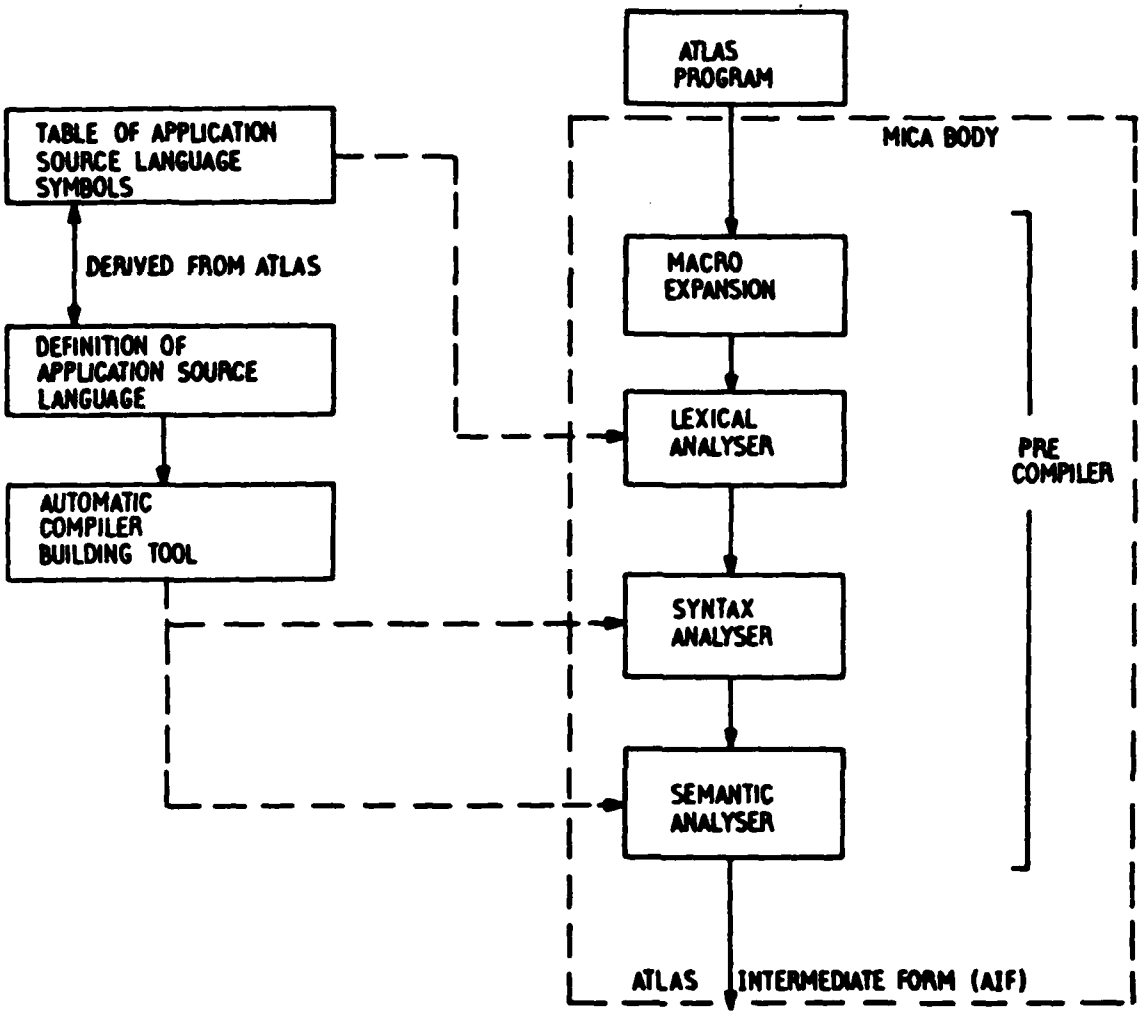


Figure 10

The AIF is in a reverse Polish notation to ease subsequent processing by the post compiler. Simply reverse Polish notation is a means of representing ATLAS by the use of operators and operands. The operands form the arguments to the following operator. For example a statement number is represented in AIF as the number which is the operand, followed by the operator ØK.

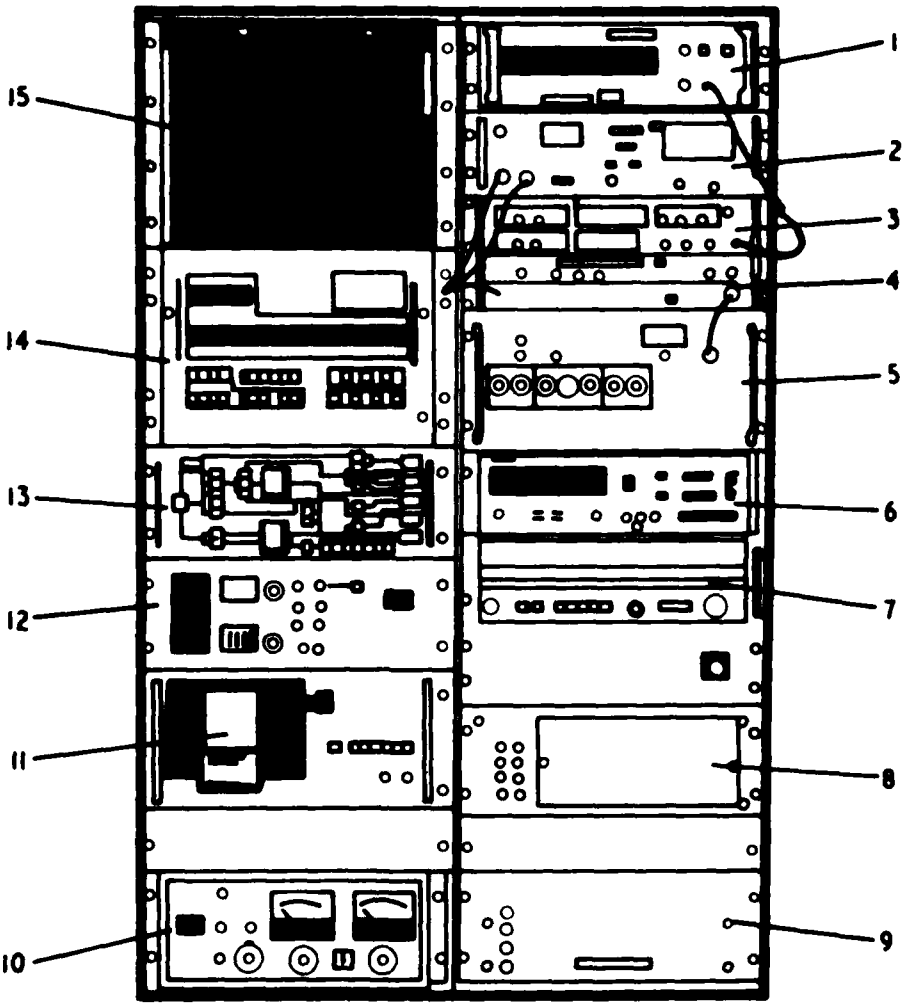
4 TALK 3 BRIEF DESCRIPTION OF THE MICA POST-COMPILER

Now moving on to the third talk in which I will describe in outline the MICA post-compiler.

The MICA post-compiler accepts as input the AIF produced by the pre-compiler. It allocates the measurement and stimuli resources of a Clansman Field Test Console to relevant test requirements, as called up in the ATLAS program

and finally generates machine code particular to the Clansman Field Test Console, shown in Figure 11.

CLANSMAN FIELD TEST CONSOLE



KEY

1 Frequency Meter
2 Frequency Analyser
3 AF Generator
4 RF Attenuator & Control
5 Synthesiser
6 AC/DC Volt meter
7 Mechanical Interface Control
8 AF Switch
9 Internal Supplies
10 EUT Power Supply
11 Printer
12 Access Unit
13 RF Switching Unit
14 Control Unit
15 Tape Reader Spooler

Alternative
Signal Generator
2 Modulation Meter
3 AF Generator
4 RF Processor
5 RF Synthesiser

Figure 11

The Mark I Clansman Field Test Console is a tape sequential ATE with stimulus and measurement instrumentation appropriate to the GO/NO GO testing of Clansman radios. The MICA post-compiler is not capable of producing code for any test envisaged, but in fact represents a subset of the ATE's abilities. For example the MICA post-compiler is not capable of generating code to drive the mechanical interface control shown at position 7.

To gain an appreciation of what processing the post-compiler performs, let us return to our example ATLAS statement shown in Figure 4. This statement implies that the ATE must be capable of measuring a frequency which has a maximum possible value of 32 MHz and a voltage maximum of 1 mV, at UUT pins A1 and A2. It is the job of the resource analyser to allocate the correct measurement device and set it to the correct range. On completion of resource allocation, the final step is to produce ATE code. Let us now look a little more closely at the post-compiler building blocks.

MICA CONSTITUENT PARTS

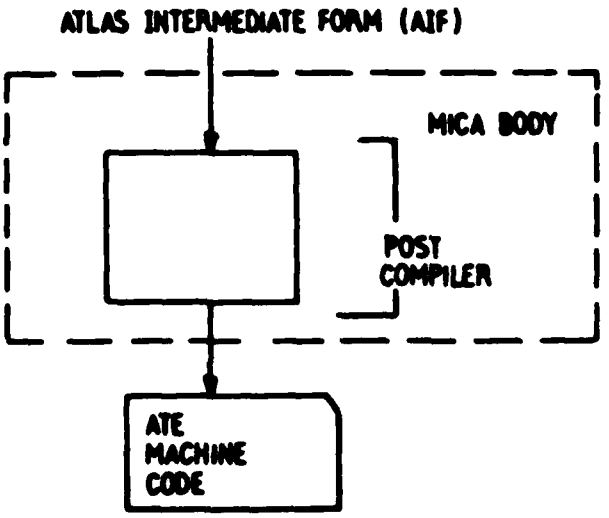


Figure 12

The first task is resource analysis, shown in Figure 13. The actual resource analyser is simply an algorithm which scans a data base describing the target ATE resources and determines which available resource, if any, is the most suitable. The data base, as shown is derived from a definition of the target ATE resources and is itself written in ATLAS.

MICA CONSTITUENT PARTS

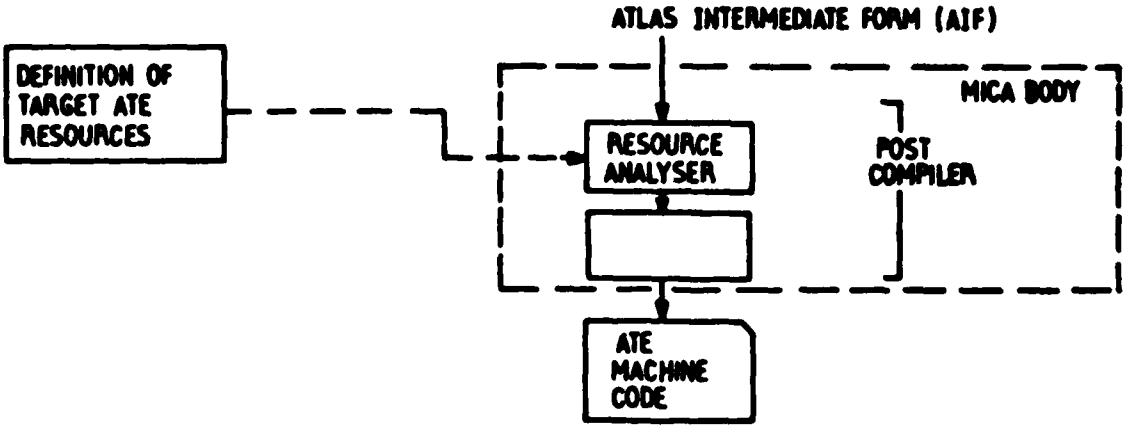


Figure 13

The final task of the post-compiler is the generation of the ATE code, shown in Figure 14. The ATE code generator consists of much special-to-application software but can be regarded as being driven by a data base derived from a description of the ATE language.

MICA CONSTITUENT PARTS

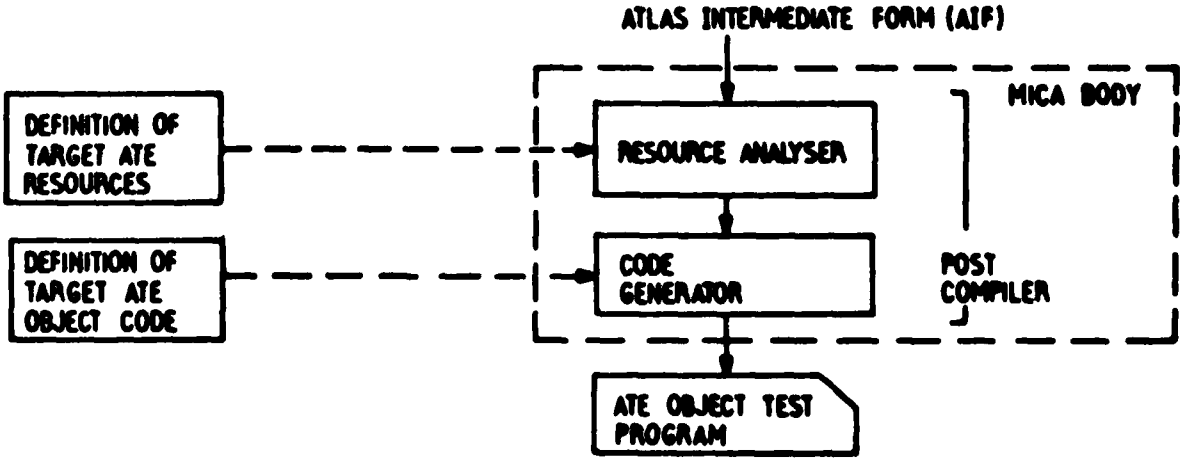


Figure 14

5 TALK 4 MICA - THE FINAL REPORT

The intention of this talk is to give you the benefit of our experienes with MICA and to emphasise that MICA is the result of research work and not development. To achieve this aim I propose to describe in outline the acceptance work which we at RSRE performed on MICA. I will then give you our opinions of MICA, based on the acceptance work and subsequent usage. Finally I will outline a slightly different approach that we have taken at RSRE, to overcome some of the problems of post-compilation.

First I will deal with the acceptance work performed on the pre-compiler.

5.1 Pre-Compiler Acceptance Work

We at RSRE performed the following acceptance tests:

i Error free source code

The pre-compiler CORAL 66 source code was compiled on a PDP 11/40 at RSRE, to determine that no CORAL 66 errors or warnings existed.

ii ATLAS grammatical analysis

RSRE constructed a number of simple error free ATLAS programs which conformed to the RSRE ATLAS subset, defined for the MICA work and submitted them to the pre-compiler to ensure that they could indeed be analysed.

iii Pre-compiler listings

The AIF produced by the pre-compiler for the compiled test programs was checked against the definition of the AIF, to ensure consistency.

iv ATLAS program error detecting ability of the pre-compiler

A known good ATLAS program had ten grammatical errors inserted into it and was input to the pre-compiler were deleted and the process repeated until all ten errors had been correctly identified and corrected.

v Pre-compiler documentation

The pre-compiler documentation was checked, on a sample basis, to ensure conformity with a slightly modified version of AVP 70 Spec 4.

5.2 Post-compiler Acceptance Work

Turning now to the acceptance work performed on the post-compiler we carried out the following checks:

i Error free source code

The post compiler CORAL 66 source code was compiled on a PDP 11/40 at RSRE to determine that no CORAL 66 errors or warnings existed.

ii Resource allocation ability

The MICA resource analyser was tested in terms of its ability to allocate Clansman Field Test Console resources to an ATLAS program which covered thirty two specification tests for the Clansman RT 353. The tests performed were of three types:

a An ATLAS program was input which called for resources that could be met by the Clansman Field Test Console and correct processing was observed.

b An ATLAS program was input which called for resources that could not be met by the Clansman Field Test Console and correct resource allocation error reports were observed.

c The resource analyser's ATLAS data base which describes the resources, was seeded with errors and correct resource allocation error reports were observed.

iii Correct code generation

The third post-compiler test was to validate that the code generator produced correct code for the Clansman Field Test Console.

iv Post-compiler documentation

The post-compiler documentation was checked, on a sample basis, to ensure conformity with a slightly modified version of AVP 70 Spec 4.

What you must appreciate is that the acceptance work that RSRE performed on MICA was very limited and only in line with the acceptance of work resulting from a research contract. By no stretch of the imagination could it be equated to full acceptance of a developed ATLAS compiler, which at a conservative estimate, would require four man years of effort.

5.3 Impressions of the Value of MICA to Industry

What then were our impressions of MICA based on the limited acceptance work performed? Our impressions are perhaps more easily expressed in terms of how well MICA meets the research aim of easing the problem associated with ATLAS compiler reconfiguration. You will remember that I identified four primary features of MICA which should permit reconfiguration. These were:

i Use of CORAL 66 to assist transportation.

ii Distinct separation between the grammatical and ATE specific processing areas.

iii Semi-automatic means of producing the pre-compiler data base.

iv Rigorous documentation.

i First the use of CORAL 66 to assist transportation

MICA was written in CORAL 66 with portability in mind. However as many of you will know CORAL 66 does not address the input/output aspects of computing and it is in this area that MICA transportation problems between different computers will arise. Another rather more serious problem comes about because of the physical size of the MICA object code. The pre-compiler is approximately 150 K words and the post-compiler is 450 K words. The pre and post compiler have been split into a number of separate tasks and each task has been further split by the use of an overlay facility such that only 32 K words is resident in core at any one time. The result of this is that the problems of transportation of MICA to a computer within the PDP 11 family are small but beyond this they can become very large indeed. By way of example a student with RSRE spent a period of six months to partially transport the pre-compiler to a GEC 4080 computer.

ii The second aspect was the distinct separation between the grammatical and ATE specific processing areas

This aim of the research has been fulfilled and appears to work very well indeed. One advantage is that it enables the pre-compiler to be used as a separate entity, as say an ATLAS program preparation tool.

iii The third aspect was the Semi-Automatic means of producing the pre-compiler data base

The use of SEMSID offers considerable advantages to the compilation of a language such as ATLAS which requires subsetting to suit particular applications. However acceptance work performed on this aspect of the pre-compiler showed that to generate a grammatical modification is certainly possible but is far from trivial.

iv The final aspect was a rigorous documentation standard

Much of RSRE's acceptance work was concentrated on documentation which is, as you will know, one of the more important aspects of software. We feel that the documentation is of a reasonable standard and provides some very useful information.

Let me now identify what we regard as the true value of MICA. First of all we believe that the MICA research work has provided a significant step forward in tackling the problems of ATLAS compilation. The philosophy and design employed is of considerable potential benefit to any future ATLAS compiler work. We believe that the pre-compiler with associated documentation is a very useful piece of general purpose software which any company using ATLAS in support of Government contracts should have access to. The pre-compiler as it stands can be used as an ATLAS programmer training aid, giving "hands on" experience or it can be used as a basis for the derivation of a complete ATLAS compiler.

Turning now to the post-compiler, the experience with which has, to our minds, shown that perhaps a general purpose post-compiler is an admirable objective but is not feasible because of the sheer size and complexity of the program to implement it. In fact we suggest that although the MICA post-compiler philosophy and design is of interest, the actual post-compiler produced is of very limited value to industry.

5.4 Experimental ATLAS to IEEE-488 Bus ATE

On this theme of the ATLAS post-compilation problem, we at RSRE have been conducting some limited in-house research into the derivation of a slightly different approach which relies on the use of the MICA pre-compiler but not the MICA post-compiler and I thought I would take this opportunity of giving a brief review of the work.

The work has resulted in an experimental implementation of an ATLAS to IEEE-488 bus ATE compiler[2] which uses the MICA pre-compiler described together with a different post-compiler approach. In Figure 15 we see the physical layout of the experimental ATE which is in essence an Intel microprocessor development system with an IEEE 488 bus controller board which we developed at RSRE. Attached to the bus we have a frequency counter, a digital multi-meter and a switching matrix. In the foreground you will see a simple UUT.

What we have done is to take advantage of the split between the pre and post compilers at the defined AIF point. In Figure 16 the hatched area shown represents the software aspects of the work which relies on an AIF processor, which is written in Pascal and runs on the microprocessor development system, to translate the AIF into a CORAL 66 program. This CORAL 66 program contains pseudo IEEE-488 bus related statements which when compiled using a standard CORAL 66 compiler are expanded by the use of a reference library.

Figure 17 gives you an idea of the appearance of the CORAL program produced by the AIF processor. The statements in the hatched area are macros which are expanded during compilation.

One advantage of this approach to post-compilation is that much of the processing is performed by commercially available software, in this case the CORAL compiler and INTEL assembler.

This now brings my short lecture on MICA to a close. I hope that it has been instructive and will make the demonstration a little more meaningful.

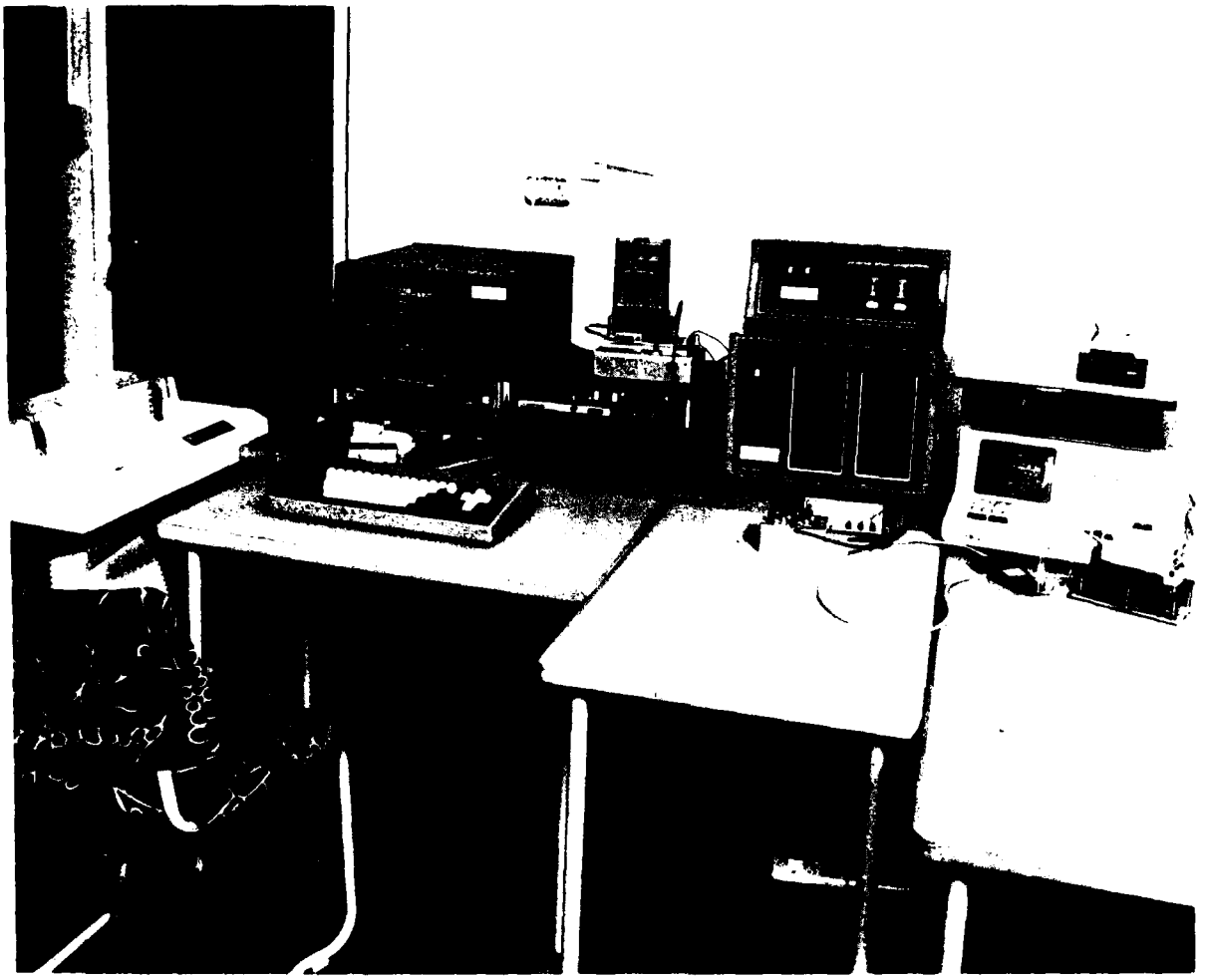


Figure 15

EXPERIMENTAL IMPLEMENTATION OF AN ATLAS TO IEEE 488 BUS ATE COMPILER

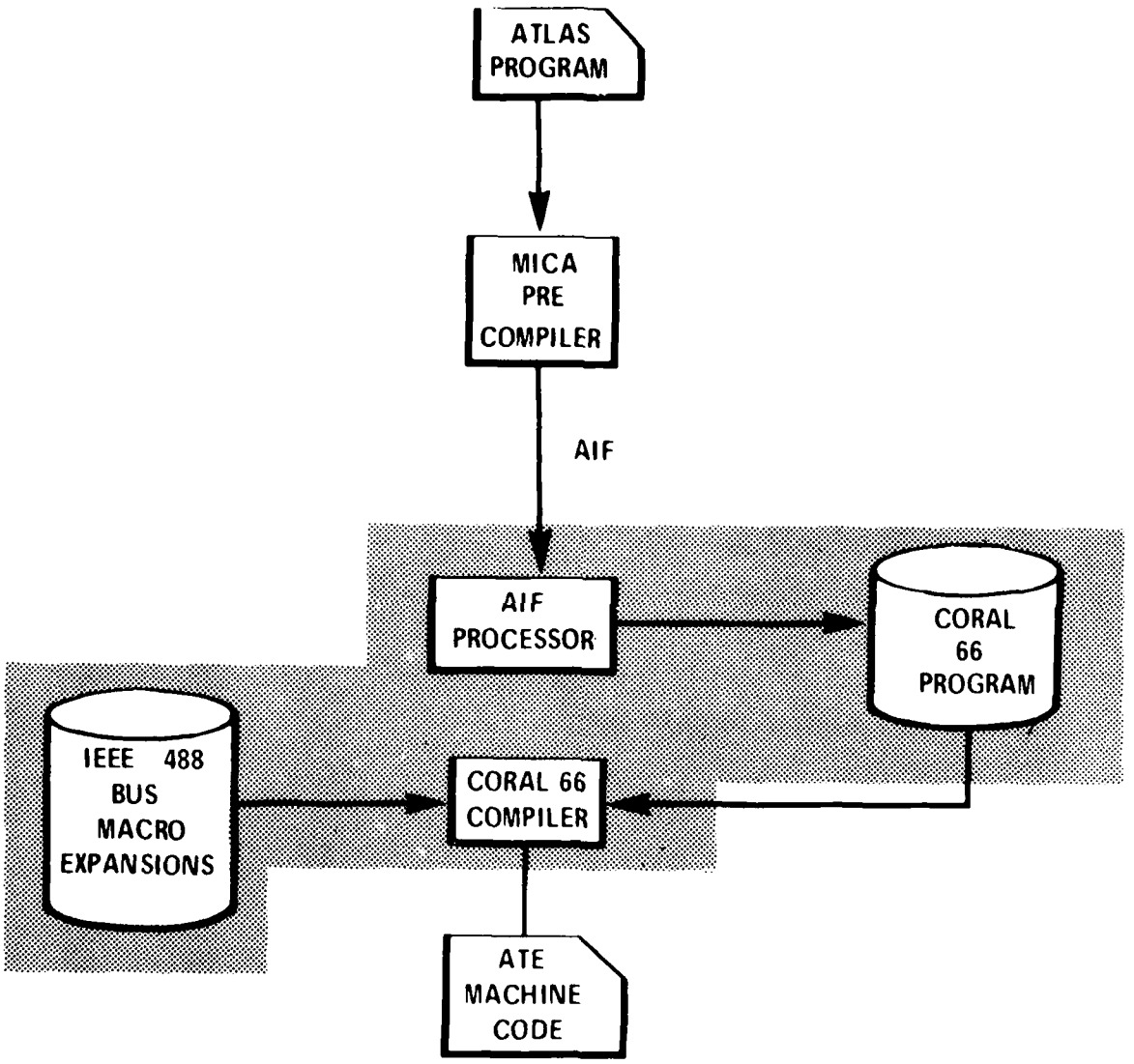


Figure 16

SAMPLE CORAL 66 PROGRAM PRODUCED BY THE THE AIF PROCESS

```
'CORAL' Test
'LIBRARY' "Inout.Ext"
'LIBRARY' ":F1:AC1.Ext"
'LIBRARY' ":F1:KLMACS"
'LIBRARY' ":F1:IEEE01.Mac"
'ABSOLUTE'(
'PROCEDURE' TTYOUT/'HEX'(FC9F) ('VALUE''BYTE');
);
'EXTERNAL'('LABEL' AC1);
'BEGIN'
'LIBRARY' ":F1:IEEE01.Set"
'BEGIN'
'LIBRARY' ":F1:IEEE01.PRC"
DECLARE;
'COMMENT'******START OF TEST PROCEDURE******;
CLEAR;
DISPLAY(''*C*LCONNECT DVM TO BUS.CONNECT RES TO BE MEASURED*C*L*
       "PRESS GO WHEN READY*C*L");
WAIT FOR MANUAL INTERVENTION:
SWITCH DVM(1);                (CONNECT DVM TO UUT VIA MATRIX)
MEASURE OHMS;                 (DVM ON AUTORANGE)
DISPLAY("*C*LMEASURED RESISTANCE=");
OUTPUT RESULT;
DISPLAY("KILOHMS*C*L");
REMOVE ALL;
'COMMENT'******END OF TEST PROCEDURE******;
'END';
'END'
'FINISH'
```

Figure 17

6 THE FORM OF THE DEMONSTRATION

A demonstration of MICA compiling an ATLAS program for the Clansman Field Console was given using a PDP 11/70. Following this the machine code produced was shown driving the Clansman Field Test Console and performing a specification test on a Clansman RT 353.

Also demonstrated was the experimental IEEE bus ATE shown in Figure 15.

REFERENCES

1 Examples in the Use of SEMSID. RSRE Memorandum No 3021 by T A D White.

2 An Experimental Implementation of ATLAS Programs on an IEEE-488 Bus Structured ATE. RSRE Memorandum No 3284 by D P Taylor.

APPENDIX A

LIST OF LEVEL ONE AND TWO MICA DOCUMENTATION HELD BY RSRE

1 LEVEL ONE

MICA pre-compiler in PDP 11 executable task form.
MICA post-compiler in PDP 11 executable task form.
ATLAS Intermediate Form (AIF) definition. Appendix 16 to document 301/3/005.
MICA RSRE subset parts 1 and 2.
MICA user's guide. Document WPL/D/434/80/137.
A code of practice for the Clansman ATLAS compiler. Document WPL/D/434/80/144.

2 LEVEL TWO

MICA pre-compiler build standard 2 version 3 software build standard specification.
MICA pre-compiler source code.
MICA post-compiler source code.
Functional requirement specification. Document 301/3/004.
MICA pre- and post-compiler design specifications. Document 301/3/005 and Document WPL/D/434/80/167.
The use of SEMSID in building syntax tables. Document WPL/D/434/80/146.
The lexical and verb and composite word tables. Document WPL/D/434/80/147.
AIF code generation. Document WPL/D/434/80/148.
RSRE research ATLAS pre-compiler software specification. Document WPL/D/434/80/150.
RSRE research ATLAS pre-compiler software specification diagrams. Document WPL/D/434/80/151.
MICA RSRE command file documentation.
Pre- and post-compiler overlay (.ODL) flow charts.
SEMSID input to pre-compiler.
MICA resource analyser AVP 70 documentation, levels 1 2 and 3 (Part 1). (PDL listings)
MICA resource analyser AVP 70 documentation, level 4. (Part II). Document WPL/D/434/79/120.
Code generator design documentation. Document WPL/D/434/80/161.